COVER SHEET

Deadline-based Escalation in Process-Aware Information Systems

Wil M.P. van der Aalst[1,2], Michael Rosemann[2], Marlon Dumas[2]

[1] Department of Information Systems

Eindhoven University of Technology, The Netherlands

w.m.p.v.d.aalst@tm.tue.nl

[2] Faculty of Information Technology

Queensland University of Technology, Australia

m.rosemann,m.dumas@qut.edu.au

Abstract

Decision making in process-aware information systems involves build-time and run-time decisions. At build-time, idealized process models are designed based on the organization's objectives, infrastructure, context, constraints, etc. At run-time, this idealized view is often broken. In particular, process models generally assume that planned activities happen within a certain period. When such assumptions are not fulfilled, users must make decisions regarding alternative arrangements to achieve the goal of completing the process within its expected timeframe or to minimize tardiness. We refer to the required decisions as *escalations*. This paper proposes a framework for escalations that draws on established principles from the workflow management field. The paper identifies and classifies a number of escalation mechanisms such as: changing the routing of work, changing the work distribution, or changing the requirements with respect to available data. A case study and a simulation experiment are used to illustrate and evaluate these mechanisms.

Keywords: Process-aware information systems, workflow management, deadline escalation, resource allocation

1 Introduction

Humans typically change their usual behavior when confronted with a deadline [12, 36]. For example, workers may skip tasks or require less information to make a decision if the cost of missing the deadline is higher than the cost or loss of service quality associated with such special actions. Such flexibility is essential in many situations yet it is rarely supported by information systems in a consistent manner.

In particular, process-aware information systems such as workflow management systems, typically do not change their behavior when confronted with deadlines. These systems stick to an idealized

model of the process even when there is no time or it is undesirable to stick to it. To address this shortcoming, this paper explores the concept of *escalation* in the context of process-aware information systems. Just like a human "escalate" (i.e., change their behavior) when unable to meet deadlines, we propose the information system to escalate in a similar fashion. Escalation may imply performing a task in a different way, allowing less qualified people to do certain tasks, or making decisions based on incomplete data.

The paper focuses on decision making in the context of deadline-based escalation, i.e., taking appropriate actions when getting close to a deadline or when it becomes clear that a deadline will not be met. The objective is to provide a decision making process and a range of relevant decision alternatives clustered according to established classifications from the field of workflow management.

As a working example, we consider the "teleclaims" process of an Australian insurance company. This process deals with the handling of inbound phone calls, whereby different types of insurance claims (household, car, etc.) are lodged over the phone. The process is supported by two separate call centers operating for two different organizational entities (Brisbane and Sydney). Both centers are similar in terms of incoming call volume (around 9000 per week), average call handling time (550 seconds), number of call center agents (90) and performance objectives (90% of all calls should be answered in less than 60 seconds). The main differences between the two centers are the underlying IT systems, the physical locations and the modes of operation (24 hrs. versus 9-5). The teleclaims process model is shown in Figure 1. The two highlighted boxes at the top show the subprocesses in both call centers. The lower part describes the process in the back-office.

This process model is expressed in terms of an Event-Process Chain (EPC) [31]. To introduce the notation let us consider the subprocess corresponding to the call center in Brisbane. The process starts with *event* "Phone call received". This event triggers *function* "Check if sufficient information is available". This function is executed by a "Call Center Agent". Then a choice is made. The circle represents a so-called *connector*. The "x" inside the connector and the two outgoing arcs indicate that it is an exclusive OR-split (XOR). The XOR connector results in event "Sufficient information is available" or event "Sufficient information is not available". In the latter case the process ends. If the information is available, the claim is registered (cf. function "Register claim" also executed by a "Call Center Agent") resulting in event "Claim is registered". The call center in Sydney has a similar subprocess and the back office process should be self-explaining after this short introduction to EPCs.

One challenge for the insurance company is dealing with an increased number of incoming phone calls during the Australian storm season (October-March). Storms cause a higher number of damages and increase the number of incoming weekly phone calls to more than 20,000. This not only puts significant burden on both call centers, but also on the succeeding back-office processes related to evaluating and managing these claims. Overtime as one way of adjusting the available resources is applied, but typically can not cope with the increased workload. Thus, the insurance company

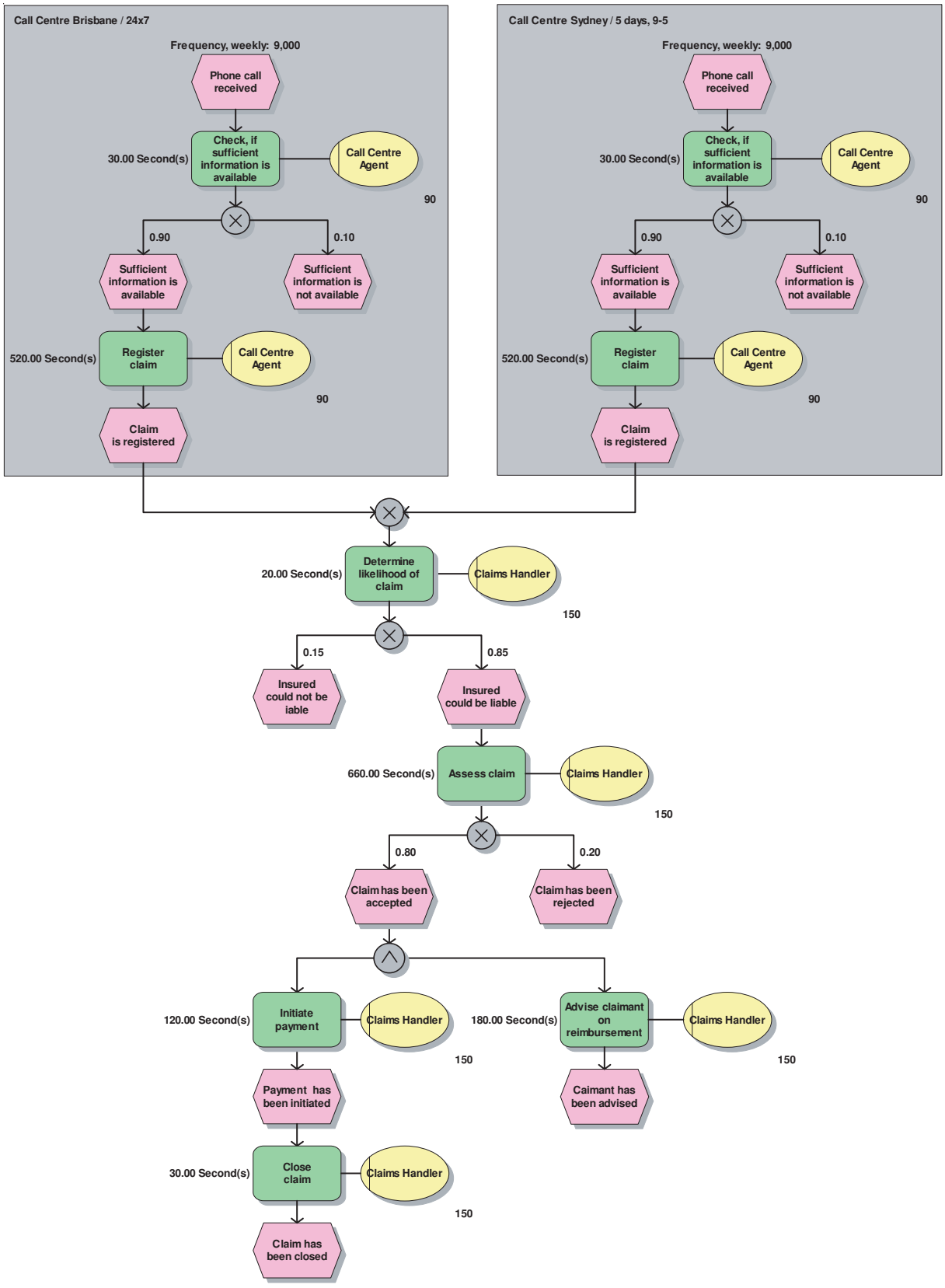

Figure 1: Insurance claim handling scenario (before escalation).

operates an "event-based response system" that differentiates four categories of situations based mainly on how severe the storms are. Based on the guidelines of this system, the first category includes localized storms and flooding and leads to a call volume of 10-50% above average for a period of at least two hours. Due to the increased call volume, customers have to wait for 5-10 minutes in the queue. The second category is triggered if strong winds, hail and structural damage occurs. This leads already to a wait time of 10-30 minutes and the call volume is 50-100% above the forecast for at least two hours. The third category covers wide-spread damage leading to waiting times of more than 30 minutes. The fourth category includes extreme cases, in which more than 80 customers would wait on the phone for more than 30 minutes.

Individual response strategies have been defined for each of these categories. The responses utilize additional external resources as well as a change in the way claims are lodged. First, additional resources are utilized through redeployment of employees from other departments (e.g. sales) and hiring of casual staff. While most of these people are trained, their performance in terms of average call handling time is lower than the performance of the permanent call center agents. Second, a streamlined way of lodging claims is applied to reduce the average call handling time and thus the waiting time in the queue. In this "rapid lodgment process", less information is collected from the claimant. This leads to an average call handling time of 380 seconds for experienced call center agents, and 450 seconds for the additionally employed agents, down from the usual average of 550 seconds. A mechanism to deal with the different performance of these two types of agents is call routing which directs all new and straight-forward cases to the additional workforce, while the more complicated follow-up calls are directed to the experienced workforce.

The four categories of situations identified by the insurance company can be seen as four levels of escalation. All levels of escalation involve the rapid lodgment process but the number of additional resources varies per level. The manager in charge for claim services together with managers in charge for the related back-office processes evaluate the severance of the weather conditions and trigger the different escalation categories. The teleclaims process model integrating the above escalation mechanisms (except for the "call routing" step) is shown in Figure 2. In the situation shown there are 30 additional call center agents and 50 additional claim handlers for the back-office process.

The four levels of escalation refer to the process as a whole. Sometimes, a single case or a limited set of cases need to be escalated. For example, Figure 2 features a situation where individual cases may be escalated based on their state. In the original process, function "Assess claim" takes on average 660 seconds, while in Figure 2 there are two functions: one taking 660 seconds and one taking only 400 seconds. This is similar to the rapid lodgment. However, the escalation does not depend upon the "global" level of escalation for the entire process. Instead it depends on how long the case in question has been open. If it has been open for more than one hour, a rapid assessment is done.

Using the teleclaims process for illustration, this paper studies the following questions:

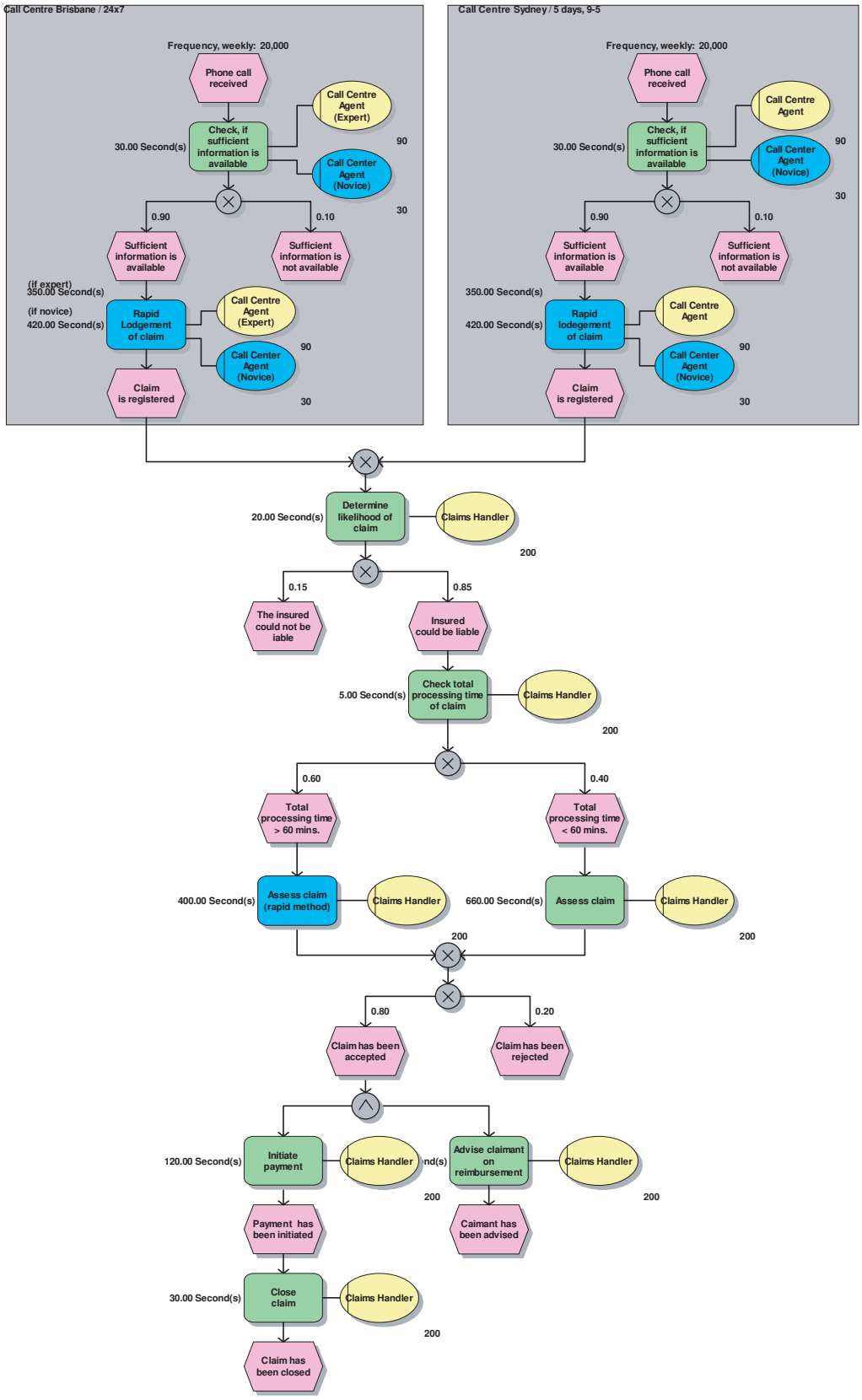

Figure 2: Insurance claim handling scenario including escalation mechanisms.

- What strategies are available to avoid or minimize tardiness (i.e. escalation strategies)?
- What is the scope of each of these escalation strategies?
- What are the tradeoffs (costs and effectiveness) related to these strategies?
- How can simulation support the selection of the level of escalation and escalation strategy?

This paper is structured as follows. The next section provides an overview of process-aware information systems and how escalations effect the main perspectives of these systems. Section 3 introduces an approach to categorize the activities involved in an escalation process. Alternative escalation mechanisms are differentiated for each of the perspectives in Section 4. Section 5 gives insights into the contributions process simulation can make for the evaluation of escalation mechanisms using the teleclaims process. Section 6 discusses the implications of our findings on current and future process-aware information systems. The paper ends with an overview of related work (Section 7) and conclusions (Section 8).

2 Process-Aware Information Systems

Process-Aware Information Systems (PAISs) support the operations of an organization based on models of both the organization and the processes involved. Traditionally, organizations have hard-coded fragments of business processes in software developed in an ad hoc manner. However, more and more organizations have started to use generic software including Workflow Management (WFM) systems, Business Process Management (BPM) systems, Enterprise Resource Planning (ERP) systems and Customer Relationship Management (CRM) systems. WFM systems such as TIBCO Staffware, Websphere MQ Workflow, or YAWL are the most typical examples of a PAIS. WFM systems "are able to interpret process definitions, interact with workflow participants and, where required, invoke [...] IT tools and applications." [20]. BPM systems extend the functionality of WFM systems beyond automation [35] into areas such as analysis, monitoring, and cross-organizational interactions. ERP systems are also process-aware: they typically support specific processes such as procurement, sales, or finance based on configurable process models. The topic of this paper is relevant to a wide range of PAISs. However, we focus on WFM systems as typical examples of PAISs.

The models of processes and organizations supported by contemporary PAISs cover four major perspectives [17]. The *process perspective* describes the control-flow, i.e., the ordering of tasks. The *data perspective* (or information perspective) describes the data that are used. The *resource perspective* describes the structure of the organization and identifies resources, roles, and groups. The *task perspective* describes individual steps in the processes and thus connects the other three perspectives.

As a preamble to classifying escalations with respect to these perspectives, we elaborate further on them using Figure 3. The figure shows four tasks: *T1*, *T2*, *T3* and *T4*. Along the *process perspective*,

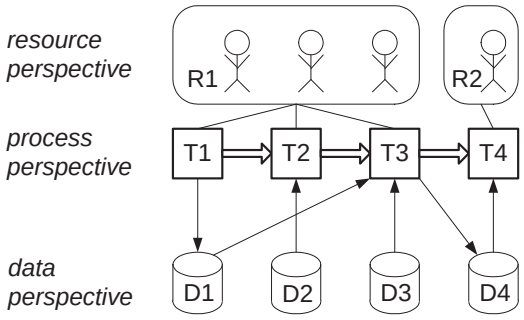

Figure 3: Illustration of the process, data, and resource perspectives.

the model in this figure has very simple control-flow dependencies: it basically states that the four tasks are executed in a sequence. Here, the process perspective describes the lifecycle of a single "case" initiated by an event, and therefore we limit ourselves to *case-driven* processes. Approving loans, processing insurance claims, billing, processing tax declarations, handling traffic violations and mortgaging, are other examples of case-driven processes. For example, a loan approval process starts with a loan application and follows a life-cycle that ends in the application being approved or rejected.

While the process perspective is concerned with the ordering of tasks, the *resource perspective* focuses on the resources needed to execute these tasks. Resources may be human (e.g. employee) or non-human (e.g. software, hardware). Non-human resources may be consumable (e.g. energy) or not (e.g. a tool). In this paper, we focus on human resources also referred to as "workers" or "users". To avoid a direct mapping of resources to tasks, resources are grouped into resource classes. Examples of resource classes are *roles*, i.e. a group of workers having similar qualifications, and *organizational units* (e.g. a department, a team, or a branch). Resources classes can be linked to tasks. For example, in Figure 3 tasks *T1*, *T2*, and *T3* can be executed by any of the three workers having role *R1*. *T4* can only be executed by the worker having role *R2*. A worker may execute multiple tasks, however for simplicity we assume that a worker cannot work on two tasks at the same time. Moreover, for a particular case each task is executed by a single worker. For example, one worker may execute *T1* for one case while another executes the same task for the next case. Resource classes may overlap, e.g. Figure 3 could be changed such that the fourth worker also has role *R1* in addition to role *R2*.

The *data perspective* is concerned with the data required to process the case. Figure 3 shows four data elements: *D1*, *D2*, *D3* and *D4*. These data elements may refer to structured (e.g. the name of a customer stored in a database) or unstructured (e.g. a Word document) data. In a PAIS data elements may be linked to tasks as shown in Figure 3. *T1* produces data element *D1*. This data element is again used by *T3*. Besides *D1*, task *T3* also uses *D3* and produces data element *D4*.

The task perspective is not shown explicitly in Figure 3. This perspective describes the content of *T1*, *T2*, *T3* and *T4*, i.e., a description of the actual work done in each of the steps.

In addition to these perspectives of PAIS, other perspectives may be relevant for escalation. First,

one could take the *context perspective* into consideration. The context describes the environment for which a process model has been designed. The initial example included two contexts: storm and non-storm season. A change in the context can motivate an escalation. Potential problems with deadlines can often be anticipated based on experiences and a sound understanding of how a specific context correlates with process performance criteria. For example, it is known that a predicted storm will cause a problem in a few hours or days. In such a scenario, it is not appropriate to wait until the first deadline-related problems occur. Second, the *goal perspective* is highly relevant to escalations since escalations are driven by the desire to meet certain goals. The *performance perspective* is closely related to the goal perspective but focuses on the actual measurement of performance indicators. This perspective includes all relevant evaluation criteria for a certain process. Escalation may be required even if there are no problems with data, resources or tasks, but instead the performance data has changed, e.g. a customer wants delivery in two days instead of four days. Yet another perspective is the *product/service perspective*. The goal of a process is to deliver a product or a service. Note that the product does not need to be a physical product. It could also be data (e.g., a decision).

In this paper we concentrate on the four "classical" perspectives (i.e., data, resource, task and process) for the following reasons. First, we use PAISs (and WFM systems in particular) as our starting point, and generally, these systems only support the specification of the four classical perspectives. Second, the context, goal, performance, and product/service perspectives can be treated as higher-level perspectives that are mapped onto the classical perspectives during system analysis and design. For example, the context perspective can be captured by the data perspective: context information (e.g., the weather) may be captured as a variable in the process. Similarly, goals and performance indicators can be made operational in terms of the classical perspectives. The product/service perspective can also be mapped onto the classical perspectives. For example, there may be the choice to offer another product because the initial product cannot be produced in time. This is an example of an escalation in the product/service perspective which can be translated into the data, resource, and process perspectives: the data changes based on the new product definition, the resources involved may change, and the process may change (e.g., another manufacturing process is followed).

3 Escalations: The 3D Approach

PAISs are usually configured on the basis of idealized models. As a result these systems have problems dealing with situations that do not conform to these models. The term *exception handling* is used to refer to activities that need to be performed when deviations appear between what is planned and what is actually happening. There is extensive literature on exception handling [6, 7, 11, 13, 14, 15, 19, 22, 30, 34] (cf. Section 7). In this paper, we focus on a particular kind of exception, namely

deadline escalation, that arises when an organization is not able to meet preset deadlines for one or more cases of a process model.

Escalations can be seen as a special family of exceptions with several distinguishing characteristics. First, while the notion of exception in general refers to a rare or exceptional situation, escalations correspond to expected events that one expects to happen regularly. Second, exceptions can be caused by all types of changes in the environment and are more difficult to identify. Meanwhile, the pre-conditions for escalation can be detected by monitoring specific metrics. For example, in this paper we focus on escalations triggered by deadline-related metrics. Finally, the handling of exceptions in general may demand non-repetitive decisions, while the decision-making behind escalations can be captured by a pre-defined set of decision alternatives. Hence, reactions to escalations can be better prepared and are often well-documented in the form of escalation procedures. In summary, while exceptions demand flexibility in dealing with unexpected situations, escalations provide the opportunity to pre-design reactions. This paper provides a set of decision alternatives for escalations triggered by the risk of missing a deadline.

We assume that each case c has a deadline D_c. If some cases do not have a deadline, we simply set the deadline to infinity, i.e., $D_c = \infty$. The deadline may be absolute or relative to the time the case started. However, the result is an absolute timestamp. Similarly, tasks may have deadlines, e.g. D_c^t is the deadline of task t for case c.[1] The completion time of a case c, denoted C_c, is the actual time the case was completed. Similarly, C_c^t is the completion time of task t for case c. Preferably, $C_c \leq D_c$ and $C_c^t \leq D_c^t$ for all cases c and all tasks t. A case c is *late* if $C_c > D_c$ and a task t is *late* for case c if $C_c^t > D_c^t$. The values of C_c and C_c^t are only known *after* the completion of the case or task. However, it is often possible to predict the completion time of a case or task. For a given case c, let P_c be the predicted completion time of c and P_c^t be the predicted completion time of task t for case c. Case c is *predicted to be late* if $P_c > D_c$ and a task t is *predicted to be late* for case c if $P_c^t > D_c^t$. Section 3.1 discusses how P_c and P_c^t could be computed.

If a case or task is late or predicted to be late, it may be desirable or even necessary to take special measures (i.e. escalations). Consider for example the reviewing process for a conference. If close to the deadline for informing the authors many reviews are still missing, the program chairs may decide to appoint additional reviewers, send reminders or base their decisions on fewer reviews. The definition of such escalations are typically not supported by contemporary PAISs. As a result, workers have to go around the system to deal with escalations. There may be several reasons for escalations, e.g. there may be seasonal fluctuations influencing the number of cases (e.g. in summer there will be more insurance claims related to fire), the number of available resources may vary (e.g. during the Christmas holidays there are not enough workers to cope with the workload), or there may

[1] The notation assumes that there are no loops, i.e. tasks can not be executed multiple times for the same case. This limitation can be lifted by assuming D_c^t to be the deadline for the last iteration of t for c.

be some kind of emergency (e.g. a catastrophe or a new law generating more work).

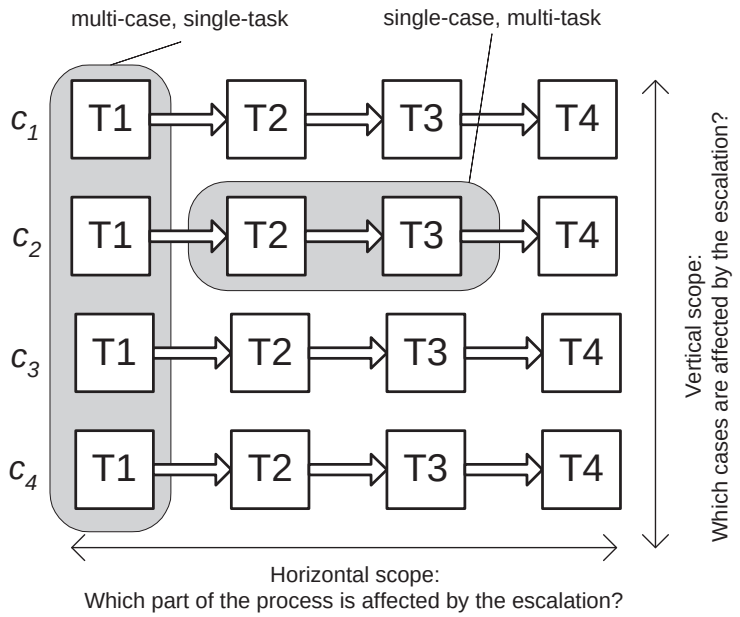

Figure 4: Scope of escalations.

Figure 4 shows that the scope of an escalation may vary in at least two dimensions. First of all, an escalation may involve a single case or multiple cases. An example of *single-case escalation* is a permit request that must be processed within a given timeframe and for which it is found, a couple of days before the deadline, that there are still several tasks to be done. On the other extreme, there is the *multi-case escalation* that involves all cases of a given process. An example is the introduction of a new law that forces organizations to handle cases within a certain time-frame. Multi-case escalation may also refer to selected cases in a given process, e.g., cases involving a claim of more than one million Euros. Second, the scope of an escalation may refer to a single task or to multiple tasks (i.e. a portion of the process). A *single-task* escalation focuses on a particular task in the process. For example, in the reviewing process of a conference an escalation may result in the skipping of a review step. On the other hand, an example where a *multi-task escalation* is relevant is the situation where the manager of the department is on holidays and (s)he is responsible for a number of tasks while no other resources are allowed to execute these tasks. As a result some tasks are late and work is piling up. An escalation mechanism may be to delegate these tasks to another person.

The teleclaims case described in the introduction proposes predominantly a multi-case escalation. A severe storm does not lead to the escalation of a single case but of an entire process. Note that in the introduction four possible escalation levels were mentioned. These refer to the whole process. However, in the back office there is also a single-case escalation: cases that are delayed more than one hour get a rapid assessment. The multi-case escalation in the teleclaims example is a single-task escalation scenario since it focuses on the initial "lodgment" task of the claim handling process.

The single-case escalation (rapid assessment for delayed cases) is also an example of a single-task escalation.

To support escalation resulting from cases and/or tasks that are late or are predicted to be late, we propose the *3D approach*: *Detect*, *Decide*, and *Do*. First, one needs to detect that there is a problem, i.e., that a case or task is (expected to be) late. Then one needs to make a decision on what to do (i.e., decide which escalation to apply). Finally, one needs to execute the escalation that was selected. There are some similarities between the 3D approach and the well-known ECA (Event-Condition-Action) rules. In fact, ECA rules have often been proposed to deal with workflow exceptions [6, 7, 13, 14, 15, 22, 30, 34]. However, the proposed 3D approach differs from the ECA rules approach in several ways. First of all, detecting whether there will be delayed cases is quite different from catching an event and evaluating a condition. Second, the decision process may involve human judgment. Finally, as will be shown in the sequel, we consider a special kind of action: *mode switching*. The various parts of a process (including all perspectives) may be in different modes. In case of an escalation, the process switches from one mode to another. We refer to this mode as an *escalation mode*. Just like the US Homeland Security Advisory System with its alert levels green (low), blue (guarded), yellow (elevated), orange (high), and red (extreme), we envision processes operating at various levels. Instead of a color code we use numbers where 0 is the normal mode of operation and higher numbers indicate modes corresponding to escalation. Consider for example task $T3$ in Figure 3. This task should be executed by a person with role $R1$ and requires data elements $D1$ and $D3$. At level 0 the PAIS enforces that $T1$ is indeed executed by a person having role $R1$ and that it can only start if both data elements $D1$ and $D3$ are available. Suppose that for a case c, task $T3$ is not executed before its deadline. This is detected and the mode is set to 1. In this mode, task $T3$ may be executed, even if $D3$ is unavailable. If this does not help, i.e., after some time $T3$ is still not executed for case c, the mode is set to 2. In mode 2, $T3$ is offered to all people having role $R3$ where $R3$ includes role $R1$. If this does not help, the mode is set to 3. In mode 3, $T3$ is simply skipped. These examples illustrate the differences between ECA rules and the 3D approach.

The remainder of this section discusses the three steps *Detect*, *Decide*, and *Do* in more detail.

3.1 Detect

The goal of deadline-based escalations is to avoid cases or tasks that are too late, i.e. $C_c > D_c$ or $C_c^t > D_c^t$, and if they happen to be too late to take rectification measures. We consider detection at the level of a single case and at the level of a process or even the whole organization. As indicated, detecting also includes monitoring the relevant context. However, this type of monitoring is outside the scope of this paper.

Let us first consider detection at the level of a single case c. There are four possible situations that result in a deadline-based escalation.

- $C_c > D_c$ or *time()*$> D_c$, i.e. the case is completed too late. (Note that we use *time()* to denote the current time.) In this situation there is not much that can be done, i.e. the case is too late anyway and only rectification measures to try and compensate for this are possible.

- $C_c^t > D_c^t$ or *time()*$> D_c^t$, i.e. task t is executed too late. In this situation, the case is delayed with respect to task t. This may be a signal to try and speed-up the case by providing additional resources.

- $P_c > D_c$, i.e. the case is predicted to complete too late and an escalation may circumvent this.

- $P_c^t > D_c^t$, i.e. it is predicted that task t is executed too late, and this may trigger some escalation.

To be able to detect this we need to have concrete values for D_c, D_c^t, C_c, C_c^t, P_c, and P_c^t. The first four values are easy to measure. The latter two estimates (P_c and P_c^t) may be calculated in various ways. Some examples:

- Based on historical information one can calculate the average time needed to execute a task (waiting time + processing time). By calculating the longest path from the current state of a case to the desired state (i.e. completion of the whole case or a specific task), we get a rough estimate for the time needed to reach that state. The "prediction engine" of Staffware [33] uses this approach to estimate the completion time of cases. A drawback of such types of approaches is that they do not take into account the actual workload, i.e. if there are many cases in the pipeline, predictions based on historical data of flow times may be too optimistic.

- Instead of using a "static" calculation based on the longest path from the current state of a case to the desired state, it is also possible to use more sophisticated techniques such as queuing analysis or simulation [5]. This is more complicated but the results will be more accurate.

- Most approaches based on queuing analysis or simulation focus on averages, i.e. the normal behavior of the flow in "steady-state". However, these techniques can also take the current state of the process (i.e. all cases) into account and do a transient analysis. For example, the current state of the process can be used to initialize a simulation model. This approach has been successfully applied using the WFM system COSA, the BPM tool Protos, and the simulation tool ExSpect [27]. It results in more accurate predictions for P_c and P_c^t but is more involved.

Factors resulting in delayed cases often do not delay a single case but multiple cases at the same time. Therefore, it may be more suitable to consider multiple cases at the same time for detection purposes. There are two ways to achieve this: (1) aggregate the results for individual cases (e.g. monitor the value of $\sum_c max((P_c - D_c), 0)$); or (2) focus on monitoring the utilization of resources relative to their capacity. The latter approach is attractive because it is relatively simple (if people are too busy, then escalate). It relies on the principle that high utilization levels usually indicate

that there is a lot of queuing and therefore this can serve as a trigger for resource-based escalations (e.g. increasing the number of staff members). It must be noted though that this approach may not detect the need for escalation in some situations. Specifically, if cases are waiting excessively long for external data, it may happen that utilization is low and yet there is a need to escalate to prevent deadline violations.

Just like the flow time of a case, the prediction of the expected utilization and the related challenges characterize this situation as one of Decision Making under Uncertainty (DMUU). DMUU is generally perceived as one of the great challenges in management science [23]. Some basic techniques, however, can be applied to consolidate relevant information and increase the quality of the decision making process in the specific context of process execution. For example, based on the routing probabilities and the number of cases arriving one can calculate the number of times each task needs to be executed. This combined with historical information about the average processing time can be used to at least estimate future utilization levels.

In Section 2 we discussed a number of perspectives on workflows, including the context perspective, the goal perspective and the performance perspective. In the general case, prediction techniques need to take these three perspectives into account. In the teleclaims scenario for example, the decision to do rapid lodgment is not based on the timing of a single case but rather on a human interpretation of the weather forecast, which is part of the context perspective.

3.2 Decide

Through the detection mechanisms just described, the cases and/or tasks that are (predicted to be) too late are identified. The detection step is followed by a decision step where the escalation measure is selected. There are three possible decision mechanisms: manual, automated, and semi-automated.

For *manual decision making*, the fact that a case or task is (predicted to be) late is forwarded to a human actor. The actor can choose from a wide range of actions as will be discussed in Section 4 (e.g. skipping a delayed task). The advantage of human judgment is that a human can take into account "fuzzy" information ranging from the weather forecast to gossip. A drawback is that the human actor may be unavailable or too busy.

Automatic decision making results in escalations without human involvement. Based on a set of rules, the right escalation is selected. For example, if the head of the department does not react within two weeks, the case is routed to her superior manager. The advantage of automated decision making is that there are no delays and using the right set of rules the quality of the decision may be high and consistent. The drawback is that rules can only detect and deal with circumstances that are clearly characterized a priori.

To combine both approaches, *semi-automated decision making* may be used, e.g. if a case or task is (predicted to be) too late, first some automatic decisions are made. If these escalations do not

help and the situation gets worse, a human may get involved. It is also possible to do it the other way around, i.e. if a case or task is (predicted to be) too late, first a human actor is notified. If this actor does not respond in time, an automated rule is applied. There is, however, the danger that a semi-automated decision making process includes shortcomings from both, the manual and the automated decision making process.

3.3 Do

The last step in the escalation process (i.e. "Do") is the actual escalation. In this phase it is important to select the most appropriate escalation strategy from a set of possible escalation alternatives. This requires a detailed evaluation of the context which demands the escalation as well as an estimation of the impact of each available escalation strategy and the related efforts. In the next section we describe possible escalation mechanisms. The intent is not to be complete but rather to provide a framework for process designers.

4 Escalation Mechanisms

An escalation is a deviation from a normal course of action. An escalation typically implements a tradeoff between, on the one hand the amount of time required to complete a task, a case, or a set of cases, and on the other hand the level of service or resource utilization. For example, an escalation may result in service degradation or resource redeployment. Different escalation mechanisms strike different tradeoffs. Accordingly, we adopt a model of escalation in which each task has an *escalation mode* and in each mode the task may behave differently with respect to the process perspective, the data perspective and the resource perspective. Below, we examine some escalation mechanisms with respect to these perspectives. For each mechanism, we provide an example and discuss its scope (single-case and/or multi-case), cost and tradeoffs. The statements regarding the cost and tradeoffs are only indicative as a high number of contextual factors will impact the actual costs or tradeoffs (e.g. Is spare capacity available?, What is the hourly rate of an alternative resource?, What penalty applies if the deadline is not met?, etc.). The proposed escalation mechanisms have been derived from related literature in the fields of process and workflow management and operations management. At this stage of our research, we have derived a first set of pre-defined escalation mechanisms and deployed some of them in case studies. Future research should explore the utilization, applicability, usefulness and completeness of these mechanisms in practice.

An overview of the strategies discussed in this section is given in Table 1.

4.1 Process Perspective

Below, we define five escalation strategies in the process perspective.

	Description	Scope	Cost
Process Perspective			
Alternative Path	An alternative task is selected or a task is skipped to achieve the deadline.	Single-case Multi-case	Deferred work / degraded quality
Escalation sub-process	A dedicated sub-process is spawned off to perform mitigation actions.	Single-case Multi-case	Sub-process dependent
Task pre-dispatching	Prepare for the execution of a task prior to completion of a previous task.	Single-case Multi-case	Discard and undo work
Overlapping	The execution of two subsequent activities overlaps.	Single-case	Additional coordination
Prioritization	Critical tasks or cases get a higher priority in order to accelerate their execution.	Multi-case	Lower-priority cases neglected
Resource Perspective			
Resource redeployment	Increase the capacity of critical resources (e.g. add more resources, overtime).	Multi-case	Additional resources
Batching	Group tasks as batch (e.g. based on location) and assign them to a single resource.	Multi-case	Batching effort
Data Perspective			
Deferred data gathering	Postpone gathering of data until the data is actually needed.	Single-case Multi-case	Deferred work
Data degradation	Tasks are allowed to be executed with less or different data.	Single-case Multi-case	Degraded quality

Table 1: Overview of proposed escalation strategies

4.1.1 Alternative path selection

Description When defining a process, one can specify that certain paths are conditional upon the potential violation of a deadline. In other words, there are different alternatives for performing a part of the process: one corresponding to the normal course, and the others corresponding to different escalation modes striking different cost tradeoffs. Two particular forms of this mechanism are: (i) *alternative task selection*, where a choice is made between executing a given task or executing an alternative (less desirable but faster) task; and (ii) *task skipping* where a choice is made between executing a task (or set of tasks) or doing nothing.

Example In the teleclaims process, the "claim lodgment" task is replaced by an alternative "rapid claim lodgment" task. This is an example of alternative task selection.

Scope This mechanism applies to both single-case and multi-case escalation.

Cost and tradeoffs This mechanism aims at speeding up the process execution by degrading the level of service or by pushing work to later stages of a process (or to other processes). A cost may be assigned to this mechanism to reflect the impact that it may have at later stages of the process. One significant tradeoff of alternative path selection is the potential negative impact on quality. There was a raison d'être in the original process model for the task and selecting an alternative path or even skipping a task is a potential risk. The opportunity costs of not executing the pre-defined task must be identified to quantify this impact. Certain cases with high quality demands might even prohibit the utilization of this escalation procedure due to its negative consequences. In addition, control mechanisms are required for cases which demand task skipping. For example, escalation procedures could demand that employees justify why a task had to be skipped. Furthermore, incentive schemes could be established which reward following the defined process model. Such control and incentive schemes are important for all types of escalations to ensure that performing the escalation procedure does not become the most popular choice. This to avoid that the intended process is crippled by its escalation mechanisms. While we acknowledge the importance of such complementary mechanisms, they will not be further discussed in the following escalation strategies.

4.1.2 Escalation sub-process

Description When it is predicted that a deadline will be violated, or when the deadline is actually violated, a sub-process is spawned off to perform actions specifically related to the deadline violation, such as notifying the appropriate stakeholders, re-negotiating a new deadline, or performing compensation actions and canceling the case. During the execution of this sub-process, the rest of the

process may be suspended. This escalation sub-process may be integrated into the process model if a significant number of process instances are expected to require escalation.

Example When a deadline violation occurs, a procedure is spawned to notify the customer. The customer is offered the choice to have the process stopped and be reimbursed, or to continue with a new deadline (which can be seen as a modification in the performance perspective).

Scope This is typically a single-case escalation mechanism, however, one can also imagine that a sub-process is spawned off to deal with multiple cases.

Cost and tradeoffs Costs depend on the nature of the escalation sub-process.

4.1.3 Task pre-dispatching

Description Under some circumstances, it is possible to start preparing for the execution of a task prior to completion of a previous task. This idea can be found in modern computer architectures, where instructions may be started before completion of preceding instructions to exploit otherwise idle resources. Two forms of pre-dispatching can be identified: pipelining and predictive branching.

In the *pipelining* mechanism, a task B that immediately follows another task A is enabled (i.e. placed in the worklist) as soon as the execution of A starts (i.e. after A has been picked from the worklist and the preparation phase for A has been completed). However, B is flagged as pre-dispatched, in such a way that whenever a resource picks this task from the worklist, it will not be allowed to proceed up to completion until A has completed, thereby ensuring that the control-flow dependency (and any underlying data dependency) is preserved.

Predictive branching applies when a decision point D that immediately follows a task C is reached. The workflow system then attempts to "guess" which branch will be taken (e.g. based on past history), and pre-dispatches the first task in the chosen branch. After completion of C, the branching condition is evaluated, leading to two possible scenarios: (i) the previously chosen branch is the one that should be taken, in which case the branch is allowed to proceed; or (ii) a different branch is taken, in which case the pre-dispatched task needs to be retracted (i.e. the task is withdrawn from the worklist, and if a resource has already picked it, the resource is notified and any preparation actions are undone).

Examples Before a clean-up team is authorized to enter an area it may be necessary to wait for approval from an inspection team, however, the preparation of the clean-up (e.g. setting up the clean-up equipment) could be pre-dispatched after a certain point in the inspection process.

Scope This mechanism applies to both single-case and multi-case escalation. The mechanism is applicable when the tasks in the process have an explicitly defined preparation phase and in the case

of predictive branching the resources must be able to undo any preparation actions.

Cost and tradeoffs The cost of this mechanism is determined by two factors: (i) heavier resource utilization as resources prepare tasks and then hold until they can start the actual execution; and (ii) in the case of failed predictive branching, the cost of undoing the preparation actions.

4.1.4 Overlapping

Description Overlapping can be applied as an escalation mechanism in the case of sequential activities. The main idea behind overlapping is that two activities can be accelerated if they are parallelized. This goes further than pipelining as the following task will be started (not just prepared) while the preceding is still processing. This is a form of alternative path selection where the choice is not to execute different activities but to weaken ordering relations between tasks.

Example A specialist in a hospital requests a sequence of tests where one request may depend on the outcome of a previous test. To speed up the process, multiple requests for tests are issued in parallel (e.g., because of a deadline). This may lead additional efforts (e.g., unnecessary tests) but shorten the flow time.

Scope This escalation mechanism applies to single case escalation.

Cost and tradeoffs The benefit of an accelerated processing of two sequential activities has to be compared with the costs related to increased coordination efforts between these two activities.

4.1.5 Prioritization

Description Higher priorities are given to certain tasks or cases, letting them overtake other cases in the consumption of resources.

Example If there are twenty customers with orders for a particular service and it is predicted that half of them will result in deadline violation, the potentially late cases are given higher priority. Similarly, a low priority could be given to cases that are already late to favor cases that may still be completed on time.

Scope This is a multi-case mechanism.

Cost and tradeoffs Giving higher priorities to some tasks or cases necessarily means lowering the priorities of others. This may result in deadline violations for certain tasks or cases that would not have occurred had the priorities been left unchanged. Apart from this, and the usual overhead on the

process execution environment of detecting, deciding and triggering the escalation procedure, the cost of this mechanism is neutral. This mechanism may be attractive in cases where the costs of being late are independent of the degree of lateness. For example, if there are several cases running late, one may choose to focus on some of them in order to meet their deadlines, and neglect the others, even if this makes these other cases violate the deadline by more time than they would otherwise have done.

4.2 Resource Perspective

Below, we define two strategies related to the resource perspective.

4.2.1 Resource redeployment

Description The idea of resource redeployment is to increase the capacity of the resources associated to cases or tasks that are running late. Resource redeployment can take many forms including: *adding more resources* (e.g. moving people between departments), *extending the scope of the roles associated with a task* (e.g. allow people with a lower role to execute the task,), *increasing the capacity per resource* (e.g. overtime) or *changing the allocation of tasks to achieve load balancing.* A special case of resource deployment is splitting, which is applied when finalizing the work for an entire batch of cases would take too long. In these cases, it might be possible to split the batch into smaller batches, which are worked on in parallel. Therefore the "Resource redeployment" mechanism can be seen as a collection of mechanisms aiming at achieving an increase in resource capacity.

Example In the teleclaims process, an escalation immediately leads to overtime being requested from the call center operators. If this is not enough, employees from the sales and service departments are redeployed to the teleclaims process, and if necessary, casual workforce is called upon.

Scope This is a multi-case escalation mechanism.

Cost and tradeoffs This strategy may lead to an increase of both variable costs (overtime and hiring additional workforce) and fixed costs (need to train people to be able to be redeployed or to train a pool of potential casual workers). Furthermore, the average performance per resource may be negatively impacted as in the teleclaims scenario. Besides increased costs there is the risk of lower quality levels. If resources start doing tasks outside of their normal routine or expertise area, the impact on quality may be negative (e.g., more errors or a less professional service to the customer). Further costs are related to the additional setup time/costs at each resource, possible additional transportation costs and the efforts related to consolidating the cases again to one batch for the next activity.

4.2.2 Batching

Description In some circumstances it may be possible to group tasks that are more efficiently treated as a batch assigned to a single resource. This can reduce the number of deadline violations when the tasks being batched are predicted to have deadline violations. For example, in location-based batching, tasks from several process instances are clustered based on their location and the location of the available resources. Task instances "close" to each other in space are executed in batch by a resource, before the resource moves to another location.

Example In the insurance claim handling process, when an event takes places at a given location (e.g. a bushfire), all on-site assessment tasks related to this event are batched and assigned to one or a group of dedicated assessors.

Scope This is a multi-case escalation mechanism.

Cost and tradeoffs Batching accelerates activities by eliminating setup times for individual activities. Costs can occur for the efforts related to batching activities.

4.3 Data perspective

Finally, we define strategies related to the data perspective.

4.3.1 Deferred data gathering

Description The gathering of certain data is postponed until the point in the process where it is actually needed. In other words, data that would normally be produced by a given task are not produced when this task completes; instead, they are gathered by the (first) task that needs them.

Example In the teleclaims scenario, a simplified claim creation form is used during escalation.

Scope Applies both to single and multi-case escalation.

Cost and tradeoffs Deferred data gathering may result in work being pushed to a later point in the process. In the teleclaims process, when the simplified claim creation form is used, some relevant data are not gathered. Some of these data may not be necessary in particular cases, and when it does become necessary, a call is made by the claim handling department to the customer. In the case of claims where an on-site assessment needs to be made, the missing data may be gathered by the assessor. Deferred data gathering can be used in conjunction with alternative path selection: In the teleclaims scenario an alternative version of the "lodge claim" task is associated to the simplified claim creation form.

4.3.2 Data degradation

Description Tasks are allowed to be executed with less or different data. For example, if a document is not available, the decision can be taken without it. It is also possible to look for other sources of less reliable or more costly data.

Example During a paper review process, the acceptance/rejection decision is usually taken on the basis of three reviews. However, if one of the reviews is missing by a given deadline, the decision may be taken with only two reviews.

Scope Applies both to single and multi-case escalation.

Cost and tradeoffs The strategy results in loss of quality of service, and in certain cases, it may result in some work being pushed to a later step in the process.

5 Simulation Study: Teleclaims Processing in the Storm Season

We now return to the teleclaims process model shown in Figure 1. To illustrate the effect of different escalations, we take this process model and evaluate different scenarios using simulation.

For our simulation study we use CPN Tools [8] which is based on Colored Petri Nets [18] as a modeling and analysis language. The reasons for using CPN Tools are its expressiveness (it is easy to model all escalation mechanisms), its theoretical basis (allowing for different types of analysis), and its simulation speed (close to a classical programming language).

Let us first simulate the process shown in Figure 1. We assume the arrival process to be Poisson (i.e., negative exponential interarrival times). Since the distribution of the call volume and number of resources over the day was not given, we assumed these to be constant over an 8 hour period per day. All activities (i.e., the functions in the EPC diagram) are assumed to have a negative exponential service time. The average service times are indicated in the diagram and so are the routing probabilities. For example, 10% of the incoming claims stop after the first step in the process. The numbers of resources were already shown in Figure 1: there are 90 call center agents in each call center and 150 claims handlers in the back office. Note that the same resource executes all steps in the process for a given claim in one of the call centers or the back office, i.e., transfer of work between resources only takes place in-between a call center and the back office.

Figure 5 shows a screenshot of CPN tools taken while simulating the top-level model and the Brisbane call center model. Figure 6 shows the CPN model of the back office. A comparison of Figure 1 (the EPC model) with figures 5 and 6 helps in understanding the more detailed CPN models. The functions in the EPC model correspond to transitions in the CPN model (represented as rectangles). The places (represented as ovals) in-between transitions denote possible states of a claim.

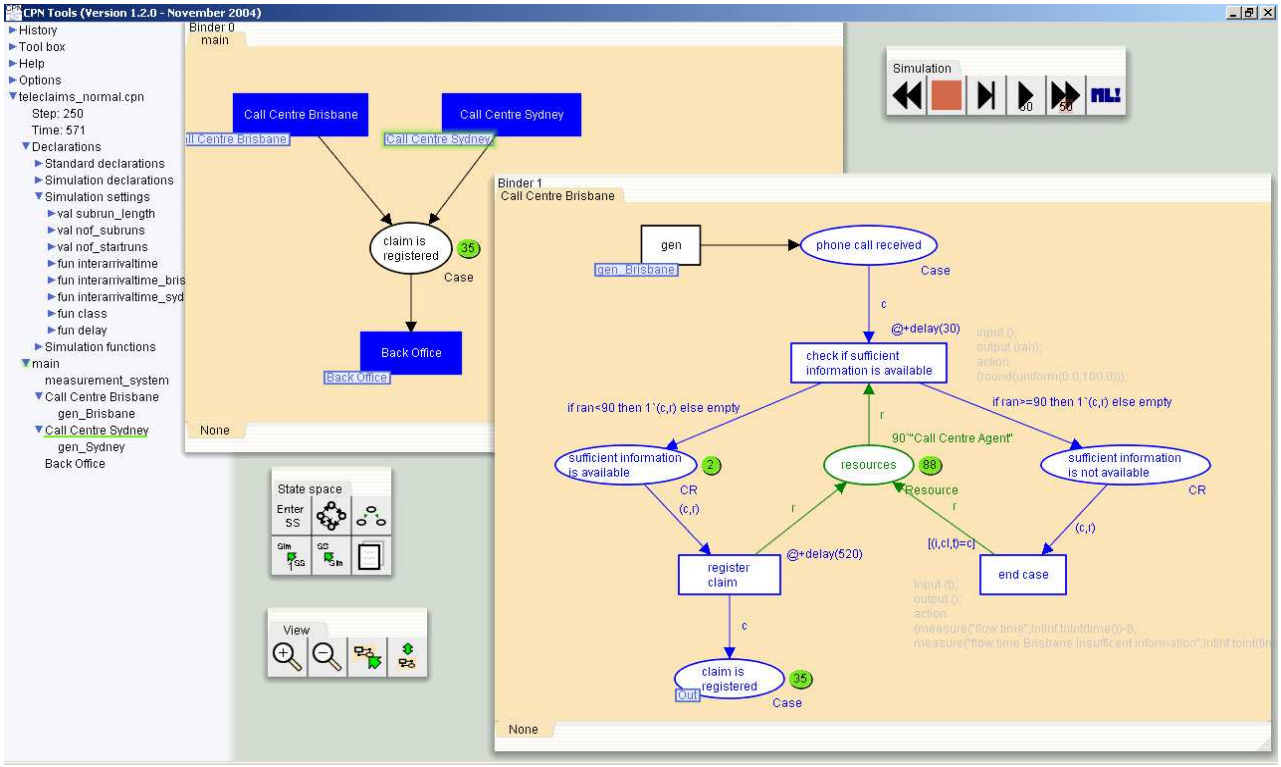

Figure 5: Screenshot of CPN showing the top-level model and the Brisbane call center.

For example, after executing "assess claim" either the state "claim has been accepted" or the state "claim has been rejected" is reached (see Figure 6). Resources are modeled by tokens in so-called resource places. For example, place "resources" in Figure 6 holds 150 tokens, each one representing a claims handler. Place "mutex" (mutual exclusion) in Figure 6 has been added to model that the tasks "initiate payment", "close claim", and "advise claimant on reimbursement" are executed by the same claim handler. Figures 5 and 6 also show the routing probabilities and refer to the delay distributions used to model the handling times. This information is needed to turn the EPC model in Figure 1 into a simulation model.

For calculating confidence intervals we assume a start run of 2 days and 10 subruns of 1 day (8 hours). The simulation shows that the average flow time is 1271 seconds, i.e., about 21 minutes. The variance of the flow time is large (i.e., a standard deviation of approximately 963 seconds). This implies that there are considerable differences between individual flow times. The average flow time of successful cases is 1667 seconds (with a 95% confidence interval of [1660,1673]). Utilization of the call center agents is 0.26 and utilization of the claims handlers is 0.60.

The process shown in Figure 1 cannot deal with the incoming claims in the storm season where the average number of claims per week goes up from 18000 claims to 40000 claims. The utilization of the claims handlers would be more than 1.0 making it is impossible to cope with the flow of work. Figure 2 shows how the insurance company can deal with 40000 claims per week. The changes

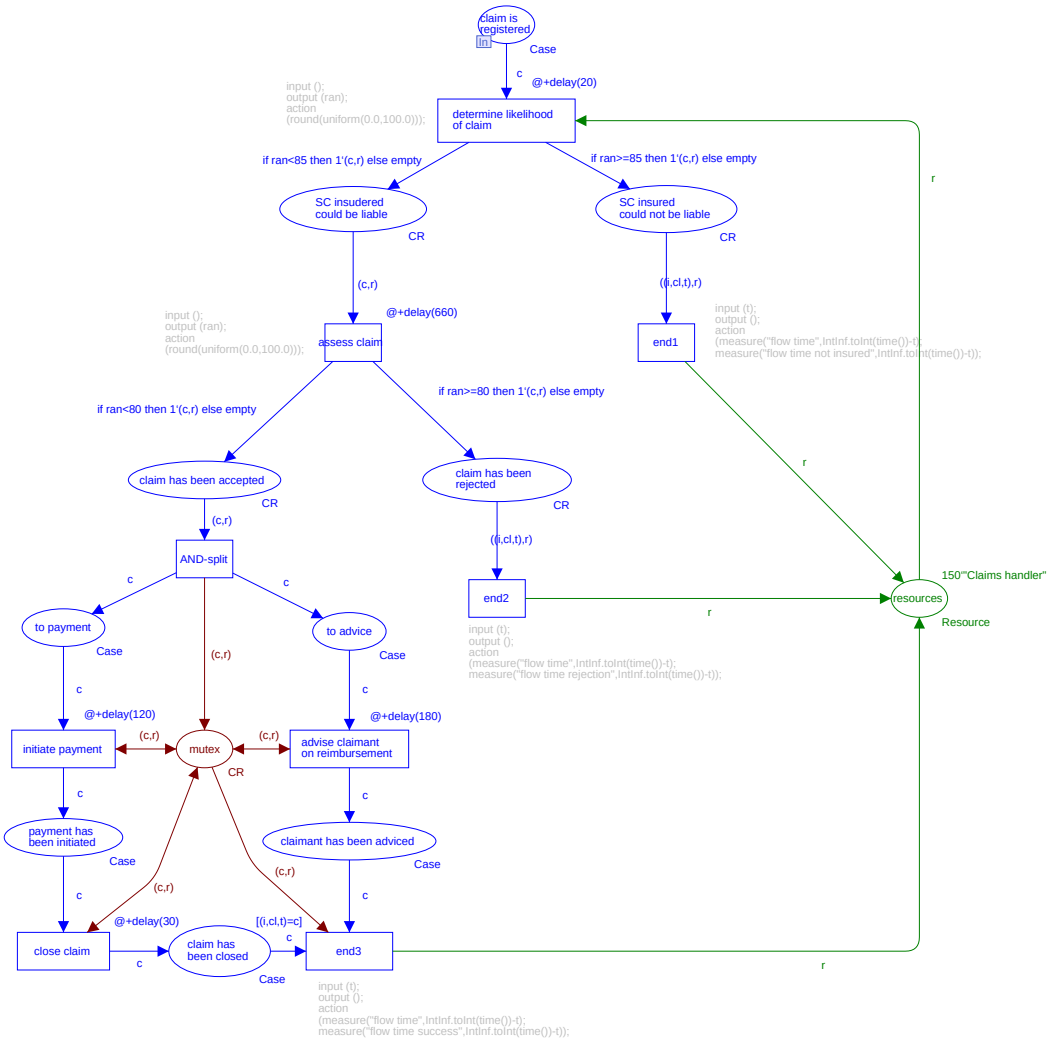

Figure 6: CPN model of the back office.

(compared to the original situation) are the addition of resources to both the call centers and the back office (resource redeployment), the rapid lodging, and the rapid claim assessment (alternative path selection). With these escalations, the organization can cope with the incoming claims and the average flow time is 937 seconds (with a 95% confidence interval of [932,943]).

6 Supporting Deadline-based Escalation in PAISs

In Section 3 we described the scope of an escalation using the *horizontal* and *vertical* dimension cf. Figure 4. In contemporary PAISs, there is generally no support for either dimension because the only way to support this is to "pollute" the normal process flow with escalations. However, it seems that single-case multi-task escalations are easier to handle than multi-case single-task escalations. The reason is that most systems offer a notion of case variables (i.e., data attached to a process instance) rather than task or process variables (i.e., data attached to a task or process and shared among instances) [29].

It is also insightful to consider the support provided by existing PAISs for each of the three phases of the 3D approach. The *Detect* phase is not directly supported by contemporary systems. Historic data and timing information may be recorded, but these data can not be referenced in the process model. Moreover, in most systems not all desired data is recorded (e.g., the queue lengths and the processing time variance) or there is no support for prediction. And even when the raw data is available, it is complicated to exploit it for prediction (see the discussion on the Staffware "prediction engine" in Section 3.1). The *Decide* phase is also difficult to support. In current systems, decisions in a workflow model represent decisions at the level of a single work-item (i.e., a task executed in the context of a case), while escalations typically span across a broader horizontal and vertical scope. It is possible to circumvent this limitation by modeling the decision process and the actual "work process" as two separate processes connected through shared variables, such that the decision process can influence the course of the work process by manipulating these shared variables. For example, one may introduce a choice in the work process that is based on an escalation variable set in the decision process. This solution impacts the maintainability negatively, since consistency must be ensured between these two processes. Other issues arise in supporting the *Do* phase. How to influence a selected scope (i.e., specific tasks and cases) and leave other parts unaffected? This can be realized by adding alternative paths in the original process. However, this way the process model becomes easily convoluted.

Let us now briefly consider the different mechanisms and discuss how to support these with current workflow management systems.

- *Process perspective: Alternative path selection.* It is relatively easy to model the selection of alternative paths or the skipping of parts of the process. However, such extensions blur the

normal workflow and they require some kind of control, e.g., through some global variable.

- *Process perspective: Escalation subprocess.* This requires the suspension of the original process. This is typically not possible. Moreover, it is difficult to exchange data and feed back the result of the escalation subprocess into the original process. For example, global data is required to reset deadlines and it is unclear how to rollback the original process based on the outcome of the escalation subprocess.

- *Process perspective: Task pre-dispatching.* Most systems consider tasks as atomic units of work. Therefore, pipelining and predictive branching are only possible by splitting up the tasks in smaller parts.

- *Process perspective: Overlapping.* Here the problems are similar to task pre-dispatching.

- *Process perspective: Prioritization.* Most systems only support simple rules like FCFS (First-Come First-Served), static priorities, or a pull mechanism (i.e., the user selects the next work-item) [28]. It is typically impossible to let this depend on the context.

- *Resource perspective: Resource redeployment.* Most workflow management systems only use static resource allocation rules. The weak support for the resource patterns [28] of contemporary systems illustrates that it is very difficult to use sophisticated rules in the allocation of work, e.g., roles are defined statically and cannot vary based on e.g. workload.

- *Resource perspective: Batching.* This is related to the piled and chained execution patterns described in [28]. Unfortunately, none of the existing systems supports these patterns.

- *Data perspective: Deferred data gathering.* Typically tasks have fixed pre- and post-conditions and it is not possible to complete tasks and start subsequent tasks without meeting the post-condition. One of the few systems that allows for this is FLOWer [3].

- *Data perspective: Data degradation.* Here the problems are similar to deferred data gathering.

The above list shows that most escalation mechanisms can be realized using existing technology, however, one needs to use workarounds that obscure the original process and require the introduction of global variables and the hiding of process logic inside applications. Therefore, we propose some extensions to facilitate escalation:

- *Views.* The system should allow for different views, e.g., it should be possible to view the process with and without the escalation mechanisms, e.g., if it is possible to skip a task when the workload is too high, then this should not be visible in the normal process.

- *Variables at multiple levels.* Most of the workflow management systems support only case variables, i.e., each case has its own set of variables. Hence it is not possible to have escalations which impact multiple cases unless some global variable is defined outside of the system. To support escalation it would be good to have variables a the task, process, and system level [29].

- *Late binding.* The concept of late binding seems to be very useful for escalation. This means that at run-time a specific model is used based on the circumstances at a specific point in time. Some systems already support the notion of process fragments, e.g., Staffware [33] allows tasks to be bound at runtime to specific subprocesses and YAWL [4] supports the use of ripple-down rules to select an appropriate process fragment. Note that late binding is not limited to the process perspective, e.g., there could be multiple organizational models that are selected based on e.g. the workload.

- *Reflection.* Current workflow management systems do not offer the concept of reflection. Reflection means that a workflow management system is aware of its processes and that it can reason about them. The goal is that this awareness is used to identify problems and modify the knowledge and models accordingly. Note that this is easier said than done. However, it may be possible to adapt mechanisms from Artificial Intelligence (AI) and agent-technology.

- *Prediction.* To detect problems that trigger escalations, it is important to be able to predict problems, e.g., based on knowledge of the process model and historical information one may want to estimate the completion time of specific cases. Currently, Staffware is one of the few workflow management systems offering a prediction feature. Unfortunately, Staffware uses fixed processing times (i.e., no stochastics) and does not take the workload into account. Clearly, prediction is not easy. It seems that, currently, simulation is the only feasible approach because times and probabilities are stochastic, there is parallel routing, and processing times depend on workload, control rules, and case variables.

In future work, we aim at extending the YAWL workflow management system [1] with escalation capabilities.[2] YAWL already supports late binding through the use of web services and the so-called worklets [4]. Features that are missing with respect to the above requirements are the ability to define multiple views, reflection, and prediction.

7 Related Work

In social sciences, researchers have investigated the effect of pressure (e.g., an approaching deadline) on the performance of people [12, 36]. These studies suggest that workers change the way of working

[2] `http://www.yawl-system.com`.

when confronted with a deadline. In most cases the effect is positive, i.e., people manage to get the work done in time. Unfortunately, current workflow management systems do not support or mimic human behavior in the presence of deadlines. A notable exception is the the FlowConnect system of Shared Web Services [16]. FlowConnect supports the definition of milestones that have planned values and actual values. These milestones are used to generate escalations and timeouts. An escalation in the context of FlowConnect means that a user is signaled that it is now critical to perform an action, i.e., the corresponding work-item is highlighted in the user's worklist. A timeout means that an action is executed if a milestone was not reached on time.

Deadline escalation can be seen as a special type of exception handling. Many proposals have been put forward to address various aspects of exception handling in workflow systems [6, 7, 13, 14, 15, 19, 22, 30, 34]. Some of these previous proposals (e.g. [6]) advocate an ECA rules-based approach, which we contrasted with the 3D approach in Section 3. Other proposals such as [13] focus on failures and rare or unexpected events rather than the ability to meet deadlines. Finally, a number of generic frameworks for structuring exception handling knowledge have been put forward. In particular, parallels can be drawn between the phases of the 3D approach (Detect, Decide and Do) and the phases for exception handling proposed by Klein & Dellarocas [19], namely "Preparing for exceptions", "Diagnosing exceptions" and "Resolving exceptions". In this respect, the 3D approach can be seen as a refinement of the general exception handling framework of Klein & Dellarocas. In the terminology of Klein & Dellarocas, what the 3D approach brings is an ontology (including resolution mechanisms) for a specific subclass of exceptions (namely deadline escalations).

Some researchers have proposed techniques addressing the specific issue of deadline escalation in workflow systems. For example, Panos & Rabinovich [24] describe an approach to dynamically adjust deadlines based on costs, expected execution times, and available slack time. In [25] this approach is refined and supported by simulation experiments. In [10, 9] the topic of capturing time constraints in workflow definitions is considered, and a PERT-like technique for the analysis of the temporal behavior of a workflow is proposed. This technique is similar to the one employed by the "prediction engine" of Staffware [33]. Other related proposals address time-related issues such as determining performance indicators based on simulation or some analytical method, but they do not consider escalations (see [26] for an overview). For example, in [37] different task prioritization policies are compared with respect to turnaround times. However, these techniques implicitly assume that processing times are independent of workload and resource capacity.

Some of the mechanisms described in this paper have been proposed by other authors. For example, the authors of [21] propose to pre-dispatch work.

Other work has focused on the scheduling of workflows [32]. Although this is highly relevant for some domains, it is not clear how this can be connected to existing WFM systems that typically aim at a high-volume of cases where routing and work-distribution decisions are taken on-the-fly.

8 Conclusion

Although organizations are forced to escalate regularly, today's (process-aware) information systems offer little support for this. In this paper, we focused on escalations triggered by the (predicted) inability to meet deadlines. Using an example of the teleclaims process of an Australian insurance company, we identified and analyzed issues related to deadline-based escalation. We then proposed a general approach to deadline-based process escalation and we presented various escalation mechanisms. The effectiveness of these mechanisms was evaluated through simulation studies [2], one of which was reported in this paper.

Future work will aim at designing and evaluating cost models for escalation. Such cost models are key during the decision phase, when the cost of applying an escalation needs to be weighed against: (1) the extent to which it decreases the probability of violating certain deadlines (or violating them to lesser degrees than without escalation); and (2) the cost of these deadline violations. On the basis of such a model, it would then be possible to answer other key questions such as: (1) when to apply a given escalation mechanism (individually); and (2) which combinations of mechanisms are most likely to work effectively together. Finally, it is necessary to design ways to seamlessly incorporate the cost model and the escalation mechanisms into existing process-aware information systems, and in particular workflow systems. Today's systems are typically unable to predict future problems and modeling the various escalation mechanisms results in spaghetti-like diagrams that cannot be maintained easily. A process modeling language that allows to clearly distinguish between a "base" process model and "escalated" versions thereof would be desirable.

Acknowledgments The third author is funded by a Queensland Government "Smart State" Fellowship co-sponsored by SAP. The authors would like to thank Greg Bird for our fruitful discussions. Thanks also to the anonymous organization that kindly provided the teleclaims case study.

References

[1] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.

[2] W.M.P. van der Aalst, M. Rosemann, and M. Dumas. Deadline-based Escalation in Process-Aware Information Systems. BPM Center Report BPM-05-05, BPMcenter.org, 2005.

[3] W.M.P. van der Aalst, M. Weske, and D. Grünbauer. Case Handling: A New Paradigm for Business Process Support. *Data and Knowledge Engineering*, 53(2):129–162, 2005.

[4] M. Adams, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Facilitating Flexibility and Dynamic Exception Handling in Workflows. In O. Belo, J. Eder, O. Pastor, and J. Falcao e Cunha, editors, *Proceedings of the CAiSE'05 Forum*, pages 45–50. FEUP, Porto, Portugal, 2005.

[5] J.A. Buzacott. Commonalities in Reengineerd Business Processes: Models and Issues. *Management Science*, 42(5):768–782, 1996.

[6] F. Casati, S. Ceri, S. Paraboschi, and G. Pozzi. Specification and Implementation of Exceptions in Workflow Management Systems. *ACM Transations on Database Systems*, 24(3):405–451, 1999.

[7] D. Chiu, Q. Li, and K. Karlapalem. A Meta Modeling Approach to Workflow Management Systems Supporting Exception Handling. *Information Systems*, 24(2):159–184, 1999.

[8] CPN Group, University of Aarhus, Denmark. CPN Tools Home Page. `http://wiki.daimi.au.dk/cpntools/`.

[9] J. Eder, E. Panagos, H. Pezewaunig, and M. Rabinovich. Time Management in Workflow Systems. In W. Abramowicz and M.E. Orlowska, editors, *Third International Conference on Business Information Systems (BIS'99)*, pages 265–280, Poznan, Polen, 1999. Springer-Verlag, Berlin.

[10] J. Eder, E. Panagos, and M. Rabinovich. Time Constraints in Workflow Systems. In M. Jarke and A. Oberweis, editors, *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE '99)*, volume 1626 of *Lecture Notes in Computer Science*, pages 286–300. Springer-Verlag, Berlin, 1999.

[11] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.

[12] J.M.P. Gevers, W. van Eerde, and C.G. Rutte. Time pressure, potency, and progress in project groups. *European Journal of Work and Organizational Psychology*, 10(2):205–221, 2001.

[13] D. Grigori, F. Casati, U. Dayal, and M.C. Shan. Improving Business Process Quality through Exception Understanding, Prediction, and Prevention. In P. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. Snodgrass, editors, *Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 159–168. Morgan Kaufmann, 2001.

[14] C. Hagen and G. Alonso. Flexible Exception Handling in the OPERA Process Support System. In *International Conference on Distributed Computing Systems*, pages 526–533, 1998.

[15] S.Y Hwang and J.Tang. Consulting Past Exceptions to Facilitate Workflow Exception Handling. *Decision Support Systems*, 37(1):49–69, 2004.

[16] A. Iordachescu. *FlowConnect: Process Timing and Distribution Concepts*. Shared Web Services, Ultimo, NSW, Australia, 2004.

[17] S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.

[18] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1*. EATCS monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1997.

[19] M. Klein and C. Dellarocas. A Knowledge-Based Approach to Handling Exceptions in Workflow Systems. *Journal of Computer-Supported Collaborative Work*, 9("3-4"):399–412, 2000.

[20] P. Lawrence, editor. *Workflow Handbook 1997, Workflow Management Coalition*. John Wiley and Sons, New York, 1997.

[21] J. Liu, S. Zhang, J. Cao, and J. Hu. An Agent Enhanced Framework to Support Pre-dispatching of Tasks in Workflow Management Systems. In Y. Han, S. Tai, and D. Wikarski, editors, *International Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002)*, volume 2480 of *Lecture Notes in Computer Science*, pages 80–89. Springer-Verlag, Berlin, 2002.

[22] Z. Luo, A. Sheth, K. Kochut, and J. Miller. Exception Handling in Workflow Systems. *Applied Intelligence*, 13(2):125–147, 2000.

[23] I. Lustig. Planning under Uncertainty (Interview with George Dantzig). `http://www.informs.org/History/dantzig/in_interview_irv10.htm`, 2005.

[24] E. Panagos and M. Rabinovich. Escalations in workflow management systems. In *Proceedings of the workshop on Databases: Active and Real Time (DART-96)*, pages 25–28. ACM Press, 1997.

[25] E. Panagos and M. Rabinovich. Reducing Escalation-Related Costs in WFMSs. In *Workflow Management Systems and Interoperability*, pages 107–127. Springer-Verlag, Berlin, 1998.

[26] H. Reijers. *Design and Control of Workflow Processes: Business Process Management for the Service Industry*, volume 2617 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2003.

[27] H.A. Reijers and W.M.P. van der Aalst. Short-Term Simulation: Bridging the Gap between Operational Control and Strategic Decision Making. In M.H. Hamza, editor, *Proceedings of the IASTED International Conference on Modelling and Simulation*, pages 417–421. IASTED/Acta Press, Anaheim, USA, 1999.

[28] N. Russell, W.M.P.van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In O. Pastor and J. Falcao e Cunha, editors, *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05)*, volume 3520 of *Lecture Notes in Computer Science*, pages 216–232. Springer-Verlag, Berlin, 2005.

[29] N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Data Patterns: Identification, Representation and Tool Support. In L. Delcambre, C. Kop, H.C. Mayr, J. Mylopoulos, and O. Pastor, editors, *24nd International Conference on Conceptual Modeling (ER 2005)*, volume 3716 of *Lecture Notes in Computer Science*, pages 353–368. Springer-Verlag, Berlin, 2005.

[30] H. Saastamoinen and G.M. White. On handling exceptions. In *Proceedings of the ACM Conference on Organizational computing systems*, pages 302–310. ACM Press, 1995.

[31] A.W. Scheer. *ARIS: Business Process Modelling*. Springer-Verlag, Berlin, 2000.

[32] P. Senkul, M. Kifer, and I.H. Toroslu. A Logical Framework for Scheduling Workflows under Resource Allocation Constraints. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 2002)*, pages 694–705, Hong Kong, China, 2002. Springer-Verlag, Berlin.

[33] Staffware. *Staffware Process Suite Version 2 – White Paper*. Staffware PLC, Maidenhead, UK, 2003.

[34] D.M. Strong and S.M. Miller. Exceptions and exception handling in computerized information processes. *ACM Transactions on Information Systems*, 13(2):206–233, 1995.

[35] M. Weske, W.M.P. van der Aalst, and H.M.W. Verbeek. Advances in Business Process Management. *Data and Knowledge Engineering*, 50(1):1–8, 2004.

[36] R.M. Yerkes and J.D. Dodson. The relation of strength of stimulus to rapidity of habit-formation. *Journal of Comparative Neurology and Psychology*, 18:459–482, 1908.

[37] J.L. Zhao and E.A. Stohr. Temporal Workflow Management in a Claim Handling System. In *Proceedings of the international joint conference on Work activities coordination and collaboration (WACC'99)*, pages 187–195. ACM, 1999.